Amendments to the Claims

Please amend claims to be as follows.

1. (currently amended) A method of providing opportunistic functional testing within a central processing unit (CPU), the method comprising:

executing a computer program having both pre-scheduled redundant and non-redundant operations on multiple functional units of a same type in the CPU;

automatically comparing outputs from the multiple functional units; and

checking results of the comparison only for the pre-scheduled redundant operations but not for the pre-scheduled non-redundant operations,

wherein the pre-scheduled redundant operations are opportunistically scheduled by a compiler ~~to take advantage of an otherwise idle functional unit during a cycle~~ during compilation of source code for the computer program and before beginning execution of the computer program to execute redundantly on at least two of the multiple functional units to utilize an otherwise idle cycle for at least one of the multiple functional units for purposes of functional testing, and

wherein multiple of the pre-scheduled non-redundant operations are scheduled by the compiler during compilation of the source code for the computer program and before beginning execution of the computer program to execute in parallel on the multiple functional units for purposes of performance.

2. (original) The method of claim 1, wherein automatically comparing the outputs from the multiple functional units is performed by comparator circuitry within the CPU that is coupled to receive the outputs.

3. (original) The method of claim 2, further comprising:

setting a comparison flag based on output of the comparator circuitry.

4. (original) The method of claim 3, wherein checking results of the comparison is performed by examining the comparison flag.

5. (original) The method of claim 4, further comprising:
if examination of the comparison flag indicates an error, then halting the execution and providing a notification of the error.

6. (canceled)

7. (currently amended) The method of ~~claim 6~~ claim 1, wherein the compiler is configured with various levels of aggressiveness with respect to scheduling of the redundant operations.

8. (original) The method of claim 7, wherein the levels of aggressiveness include levels more aggressive than just taking advantage of otherwise idle functional units.

9. (original) The method of claim 8, wherein a high level of aggressiveness forces all operations on a functional unit to be performed redundantly on another functional unit of the same type.

10. (original) The method of claim 1, wherein the functional units comprise floating point units.

11. (original) The method of claim 1, wherein the functional units comprise arithmetic logic units.

12. (currently amended) A microprocessor with built-in functional testing capability which is controllable per execution cycle, the microprocessor comprising:

multiple functional units of a same type;

registers that receive outputs from the multiple functional units; and

comparator circuitry that also receives the outputs from the multiple functional units and compares the outputs to provide functional testing during pre-scheduled redundant operations but not during pre-scheduled non-redundant operations,

wherein the pre-scheduled redundant operations are opportunistically scheduled by a compiler during compilation of source code for a computer program and before beginning execution of the computer program to execute redundantly on at least two of the multiple functional units to utilize an otherwise idle cycle for at least one of the multiple functional units for purposes of functional testing, and

wherein multiple of the pre-scheduled non-redundant operations are scheduled by the compiler during compilation of the source code for the computer program and before beginning execution of the computer program to execute in parallel on the multiple functional units for purposes of performance.

13. (original) The microprocessor of claim 12, wherein the multiple functional units comprise floating point units.

14. (original) The microprocessor of claim 12, wherein the multiple functional units comprise arithmetic logic units.

15. (canceled)

16. (original) The microprocessor of claim 12, further comprising:
    at least one flag coupled to receive results from the comparator circuitry.

17. (original) The microprocessor of claim 16, wherein the flag is ignored if different operations are performed on the multiple functional units and is checked if a same redundant operation is performed on the multiple functional units.

18. (canceled)

19. (currently amended) A computer-readable program product stored on a computer-readable medium for execution on a target microprocessor with multiple functional units of a same type, the program product comprising executable code that includes a redundant operation pre-scheduled by a compiler during compilation of source code to generate the computer-readable program product and before beginning execution of the computer-readable program product, wherein the redundant operation is pre-scheduled to execute in parallel on two functional units ~~to take advantage of one of the functional units that would otherwise be idle during a cycle~~ to utilize an otherwise idle cycle for one of the functional units for purposes of functional testing, wherein the program product is configured to execute on a microprocessor having comparator circuitry to automatically compare outputs of the two functional units for the pre-scheduled redundant operation but not for pre-scheduled non-redundant operations.

20. (currently amended) An apparatus for providing opportunistic functional testing within a CPU, the apparatus comprising:
    means for executing a computer program having both pre-scheduled redundant and pre-scheduled non-redundant operations on multiple functional units of a same type in the CPU;

means for automatically comparing outputs from the multiple functional units; and

means for checking results of the comparison only for the pre-scheduled redundant operations but not for the pre-scheduled non-redundant operations,

wherein the pre-scheduled redundant operations are opportunistically scheduled by a compiler ~~to take advantage of an otherwise idle functional unit during a cycle~~ during compilation of source code for the computer program and before beginning execution of the computer program to execute redundantly on at least two of the multiple functional units to utilize an otherwise idle cycle for at least one of the multiple functional units for purposes of functional testing, and

wherein multiple of the pre-scheduled non-redundant operations are scheduled by the compiler during compilation of the source code for the computer program and before beginning execution of the computer program to execute in parallel on the multiple functional units for purposes of performance.